IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appellant:	David H. Lin	Patent Application	
Application No.:	10/823,845	Group Art Unit:	2166
Filed:	April 14, 2004	Examiner:	Ahluwalia, Navneet K.

For: Method and Apparatus for Multi-process Access to a Linked List

REPLY BRIEF

In response to the Examiner's Answer mailed on October 28, 2010, Appellant respectfully submits the following remarks.

REMARKS

Appellant is submitting the following remarks in response to the Examiner's Answer. In these remarks, Appellant is addressing certain arguments presented in the Examiner's Answer. While only certain arguments are addressed in this Reply Brief, this should not be construed that Appellant agrees with the other arguments presented in the Examiner's Answer.

Examiner's Answer page 17 line 1 to page 18 line 18

The Examiner's Answer appears to assert that Gao teaches "marking the subsequent element in the linked-list as in-use when a breakpoint is encountered" at Figure 3, Col. 3 lines 39-67, Col. 4 lines 36-40, Col. 4 lines 62-67 and Figures 5A and 5B. Gao states at Col. 3 lines 39-67, which refers to Figure 3,

> FIG. 3 shows a container for a queue according to the preferred embodiment of the invention implemented on the computer system of FIG. 1. In FIG. 3, container 305 includes in-use and data valid flags 310 and 315, data field 320, and next pointer 325. In-use flag 310 indicates whether the container is being used at the current time (in other words, whether or not the container is available for use). In use flag 310 is the preferred embodiment for a container lock, which allows only one client to use a container at a time. Data valid flag 310 indicate whether a container holds valid data. But data valid flag 315 is not absolutely required, and can be part of data field 320.
>
> Returning to FIG. 3, data field 320 stores the data in the container. The type of data stored in the container is generally not limited, although queues designed to store specific types of data are possible. Similarly, the amount of data stored in the container is generally not limited. Finally next pointer 325 points to the next container in the queue (or, if the queue has an end and the current container is the last container in the queue, next pointer 325 is a null pointer).
>
> FIG. 3 shows only the elements of container 305 in the preferred embodiment. A more complex container can optionally contain some attributes (such as type, length, and priority) of the data the container contains.
>
> FIGS. 4A and 4B show a queue constructed from the queue head of FIG. 2 and multiple containers of FIG. 3 before and after accessing one of the containers on the computer system of FIG. 1.

Although Gao teaches at Col. 3 lines 39-67 and Figure 3 an in-use flag that allows only one client to use a container at a time, Gao does not teach or suggest "marking the subsequent element in the linked-list as in-use when a breakpoint is encountered."

Appellants further note that Gao does not teach using Gao's next pointer to point at Col. 3 lines 39-67 to a subsequent element in the linked-list for the purpose of marking the subsequent element as in-use when a breakpoint is encountered.

Gao states at Col. 4 lines 36-40,

> At step 510, the client locates a container in the queue. At step 515, attempts to lock the container, so that no other client can use the container. IN the preferred embodiment, an atomic set and swap operation is used to try to lock the container by setting the in-use flag to 1.

Although Gao teaches setting the in-use flag using an atomic operation at Col. 4 lines 36-40 so that no other client can use the container, Gao does not teach or suggest "marking the subsequent element in the linked-list as in-use when a breakpoint is encountered," at Col. 4 lines 36-40.

Gao states at Col. 4 lines 62-67,

> If the client is looking for an empty container and the selected container has data, or if the client is looking for a container with data and the selected container is empty, then at step 530 the client unlocks the container and returns to step 510 (FIG. 5A). Otherwise, at step 535 the client uses the container as desired.

Therefore, Appellant understands Gao to teach unlocking a container if it does not satisfy the client's search criteria. In other words, if the client is searching for a container that has data but the locked container is empty, it unlocks the container. Similarly, if the client is searching for an empty container and the locked container has data, it unlocks the container. Otherwise, if the locked container does satisfy the client's search criteria, at step 535, the client uses the container as desired. Appellant respectfully submits that unlocking a container that does not satisfy the client's search criteria and using a container that does satisfy the client's search criteria does not teach or suggest "marking the subsequent element in the linked-list as in-use when a breakpoint is encountered."

For similar reasons, Gao does not teach or suggest "marking the subsequent element in the linked-list as in-use when a breakpoint is encountered" at Figures 5A and 5B using steps 510 and 525.

Examiner's Answer page 18 line 18 to page 19 line 2

The Examiner's Answer appears to assert that Gao teaches "creating a recommencement reference to the subsequent element," at Col. 2 lines 46-48, Col. 3 lines 9-16 and Col. 3 lines 56-59.

Gao states at Col. 2 lines 46-48, "FIG. 2 shows a queue head for a queue according to the preferred embodiment of the invention implemented on the computer system of FIG. 1." Appellant fully submits that a queue, as taught at Col. 2 lines 46-48, does not teach or suggest "creating a recommencement reference to the subsequent element."

Gao states at Col. 3 lines 9-16,

> Returning to FIG. 2, next pointer 220 points to one of the containers in the queue. Note that it does not matter which container in the queue next pointer 220 points to, so long as containers are accessible. Thus, the organization of the queue is not relevant to the invention, and the invention is equally applicable to different queue implementations. For example, the queue can be structured as a singly linked list, a doubly linked list, a circular list, or an array.

Appellant respectfully submits that teaching that the queue can be implemented as a singly linked list, a doubly linked list, a circular list, or an array does not teach or suggest "creating a recommencement reference to the subsequent element."

Gao states at Col. 3 lines 56-59, "Finally next pointer 325 points to the next container in the queue (or, if the queue has an end and the current container is the last container in the queue, next pointer 325 is a null pointer)." Applicants respectfully submit that teaching a next pointer that points to the next container in the queue does not teach or suggest "creating a recommencement reference to the subsequent element."

Examiner's Answer page 19 lines 4-11

The Examiner's Answer appears to assert that Gao teaches "updating a recommencement reference to the element to refer to a data element that is subsequent to the data element to be deleted when the element in is in-use," at Table 14 at Col. 5 lines 10-21.

First, Appellant respectfully submits that Table 14 is a typo and should read Table 1. Gao states with respect to Table 1 at Col. 5 lines 18-19, "Table 1 shows pseudo-code of the process for removing data from a container not currently in use…" Appellant respectfully submits that removing data from a container not currently in use teaches away from "when the element in is in use," as recited. Therefore, Gao does not teach or suggest "updating a recommencement reference to the element to refer to a data element that is subsequent to the data element to be deleted when the element in is in-use," at Table 1 (mistyped as Table 14) at Col. 5 lines 10-21.

Examiner's Answer page 19 lines 11-17

The Examiner's Answer appears to assert that Marcotte teaches a condition for relinquishment of control at figure 2 and Col. 7 lines 26-65.

The Examiner's Answer refers to "relinquishing of control as a requirement as explicitly being claimed in the definition of breakpoint". Appellant notes that "relinquishing of control as a requirement" is not explicitly claimed. The instant application does state that "a breakpoint definition is used to define when a first process…is required to relinquish control over the linked-list so that a second process can gain access to the linked list." (Paragraph 10, emphasis added.) In addition, the instant application gives examples of how a breakpoint can be defined: establishing a maximum number of data elements that can be traversed in a single access session; and establishing a maximum time limit for a single access session. Appellant now assumes (for the sake of argument only, which is not an admission on the part of the Appellant) that the claim construction apparently used by the Office Action is correct:

that the combination must teach a process that marks the subsequent element in the linked-list as in-use when that process is required to relinquish control to another process.

As acknowledged by the Office Action (p. 4), Gao does not teach "relinquishing of control as a requirement [of the marking action]". Marcotte is directed to the use of an exclusive lock to protect a data structure. The Office Action alleges (p. 4) that FIG. 2 and Col. 7 lines 26-65 of Marcotte discloses "how the lock that is hel[d] can be in a priority state and thus the queue would have to wait, similarly if the queue that was requiring the lock was in priority state the current item would have to relinquish the lock". Appellant disagrees with this characterization of the reference. The only discussion of priority in Marcotte is the following: "Referring to FIG. 2, state 101 includes a 32 bit word including E 121, W 122, L 123, P 124 and R 125 values, where…a bit in field L 123 indicates that the lock is held in a higher priority state…" Appellant submits that this brief mention of priority says nothing about waiting, and certainly does not teach that a task would be required to wait when the bit in field L 123 indicated that the lock is in a high priority state. Appellant submits that the Office Action's reading of this passage in Marcotte is beyond both the literal and implicit teachings of the reference. Rather than providing any reasoning or evidentiary foundation as to why a person of ordinary skill in the art would understand a lock held in a higher priority state to be the same as being the condition of being required to relinquish the lock, the Office Action has instead merely came to the conclusion that the two are the same.

The relied-upon passage in Marcotte also teaches one task waiting on a lock that is held by another task (Marcotte, Col. 7 lines 35-55). Appellant assumes that a person of ordinary skill in the art would also understand this passage to teach, implicitly, that the lock holder eventually relinquishes control to the waiter. Even so, this is not the same condition in which one task is required to relinquish control to another task. Furthermore, even assuming that the Office Action's characterization of Marcotte is accurate, so that Marcotte does teach that some tasks are required to relinquish under

some conditions, Marcotte does not teach or suggest the specific feature of detecting and acting on this required-to-relinquish condition, as included in Claim 1.

CONCLUSION

In view of the above remarks, Appellant continues to assert that the combination of Gao and Marcotte does not describe, teach or suggest Claims 1-22, for reasons presented above and for reasons previously presented in the Appeal Brief.

Respectfully submitted,

WAGNER BLECHER LLP

Dated: 12/27/2010

/John P. Wagner, Jr./
John P. Wagner, Jr.
Registration Number: 35,398

WAGNER BLECHER LLP
123 Westridge Drive
Watsonville, CA 95076
(408) 377-0500